WHAT IS CLAIMED IS:

Claim 1:

A compiler for converting source code for a program written in a programming language into object code in a machine language, comprising:

an optimization execution unit for performing an optimization process for an object program written in a machine language; and

a program modification unit for modifying said object program in order to absorb a difference in content between the point of origin of an exception process, which occurs in response to the execution of a command in said object program, and a location whereat said exception process is performed.

Claim 2:

The compiler according to claim 1, wherein, if there is a difference in content between the point of origin of an exception process, which occurs in response to the execution of a command in said object program, and a location whereat said exception process is performed, said program modification unit generates compensation code to compensate for said difference, and inserts said compensation code into said object program.

Claim 3:

The compiler according to claim 1, wherein said program modification unit includes:

a pre-processor for, before said optimization execution unit performs said optimization process, employing a Try node to examine a command that may cause an exception process in said object program to determine whether an exception process has occurred, and a Catch node for performing an inherent process when it is found an exception process has occurred; and

a post-processor for examining, in said object program that has been optimized by said optimization execution unit, said command that may cause an exception process to determine whether a difference in content exists between said command that may cause said exception process and a location whereat said exception process is performed, and for, when a difference exists, generating in said Catch node a compensation code, to be used to compensate for said difference, and a code for, after said compensation code is obtained, moving program control to said location whereat said exception process is performed.

Claim 4:

The compiler according to claim 1, wherein, before said optimization execution unit performs said optimization

process in said object program, said program modification unit divides said command that may cause an exception process into a command portion for determining whether an exception process has occurred, and a command portion for actually causing an exception process; and wherein, when an exception process occurs, said program modification unit modifies said object program to shift program control to said command portion that actually caused said exception process.

Claim 5:

The compiler according to claim 1,

wherein, when in said object program said command that may cause an exception process consists of a command portion for determining whether an exception process has occurred and a command portion for actually causing an exception process, before said optimization execution unit performs said optimization process, said program modification unit divides said command in said object program that may cause said exception program into said command portion for determining whether an exception process has occurred and said command portion for actually causing an exception process;

wherein, when an exception process actually occurs, said program modification unit modifies said object program in order to shift program control to said command portion that actually caused said exception process;

wherein, when an exception process has not occurred, said optimization execution unit, before performing said optimization process, establishes a Try node for examining said command that may cause an exception process to determine whether an exception process has occurred, and a Catch node for performing an inherent process when an exception process has occurred;

wherein, in said object program that has been optimized by said optimization execution unit, said program modification unit examines said command that may cause an exception process to determine whether a difference in content exists between said command that may cause said exception process and said location whereat said exception process is performed; and

wherein, when a difference exists, said program modification unit generates, in said Catch node, a compensation code for compensating for said difference and, after said compensation code is obtained, a code for moving program control to said location whereat said exception process was performed.

Claim 6:

A computer system comprising a compiler for converting program source code written in a programming language into object code written in a machine language, wherein said compiler includes:

an optimization execution unit for performing an

optimization process for an object program written in a machine language; and

a program modification unit for modifying said object program in order to absorb a difference in content between the point of origin of an exception process, which occurs in response to the execution of a command in said object program that may cause said exception process, and a location whereat said exception process is performed.

Claim 7:

An optimization method for optimizing a program to increase processing efficiency comprising the steps of:

preparing a basic block that includes a portion for examining, in an object program, a command that may cause an exception process to determine whether an exception process has actually occurred, and that includes a command for, when an exception process has occurred, moving program control to a portion whereat said exception process is performed; and

generating in said basic block compensation code, when a difference in content exists between the point of origin of said exception process and said portion whereat said exception process is performed, for compensating for said difference.

Claim 8:

An optimization method for optimizing a program to increase processing efficiency comprises the steps of:

establishing a basic block that includes a Try node for examining, in an object program, a command that may cause an exception process to determine whether an exception process has occurred, and a Catch node for performing an inherent process when an exception process has occurred;

performing optimization of said object program wherein said basic block is established;

examining said optimized object program to determine whether a difference in content exists between said command that may cause an exception process and a location whereat said exception process is performed; and

generating in said Catch node of said basic block when a difference exists, a compensation code for compensating for said difference, and a code for, after said compensation code has been obtained, moving program control to said location whereat said exception process is performed.

Claim 9:

The optimization method according to claim 8, wherein said step of determining whether a difference in content exists between said command that may cause an exception process and said location whereat said exception process is performed includes a step of:

removing said basic block prepared for said command that may cause an exception process when no difference in content exists.

Claim 10:

An optimization method for optimizing a program to increase processing efficiency comprising the steps of:

dividing code, in an object program, that may cause an exception process into code for determining whether an exception process has occurred and code for actually causing an exception process;

specifying said code obtained at said division step as branches of a control flow graph;

designing said control flow graph so that when an exception process occurs, program control is shifted to said code that actually caused said exception process; and

performing said optimization process for said object program that has been modified.

Claim 11:

The optimization method according to claim 10, further comprising the steps of:

determining whether code for compensating for a difference in content between the point of origin of an exception process and code for actually causing said

exception process have been generated in a block that includes code for the actual performance of said exception process after the optimization process has been run; and

using said control flow graphs, when said code for compensating for said content difference is not generated, for synthesizing said two code sets to obtain said code arrangement that existed before said code was divided.

Claim 12:

A computer program that permits a computer to optimize an object program comprising:

a process for preparing a basic block that includes a portion for examining a command, in an object program, that may cause an exception process, in order to determine whether an exception process has occurred, and that includes a command for, when an exception process has occurred, moving program control to a portion whereat said exception process is performed; and

a process for, when a difference in content exists between the point of origin of said exception process and said portion whereat said exception process is performed, generating in said basic block compensation code for compensating for said difference.

Claim 13:

A computer executable program comprising:

a function for examining a command, in said computer program, that may cause an exception process, to determine whether an exception process has occurred;

a function for running an exception process; and

a function for, when an exception process has occurred, shifting program control to a portion whereat said exception process is run, and for, when a difference in content exists between the point of origin of said exception process and said portion whereat said exception process is run, compensating for said difference before program control is shifted to said portion whereat said exception process is run.

Claim 14:

The computer executable program according to claim 13, wherein said function for determining whether an exception process has occurred is provided by a Try node of a Try-Catch block or by a condition branch.

Claim 15:

A storage medium, on which input means of a computer stores a computer-readable program that permits said computer to

perform:

a process for preparing a basic block that includes a portion for examining a command, in an object program, that may cause an exception process, in order to determine whether an exception process has occurred, and that includes a command for, when an exception process has occurred, moving program control to a portion whereat said exception process is performed; and

a process for, when a difference in content exists between the point of origin of said exception process and said portion whereat said exception process is performed, generating in said basic block compensation code for compensating for said difference.

Claim 16:

A program transmission apparatus comprising:

storage means for storing a program that permits a computer to perform

a process for preparing a basic block that includes a portion for examining a command, in an object program, that may cause an exception process, in order to determine whether an exception process has occurred, and that includes a command for, when an exception process has occurred, moving program control to a portion whereat said exception process is performed, and

a process for, when a difference in content exists between the point of origin of said exception process and

said portion whereat said exception process is performed, generating in said basic block compensation code for compensating for said difference; and

transmission means for reading said program from said storage means and for transmitting said program.